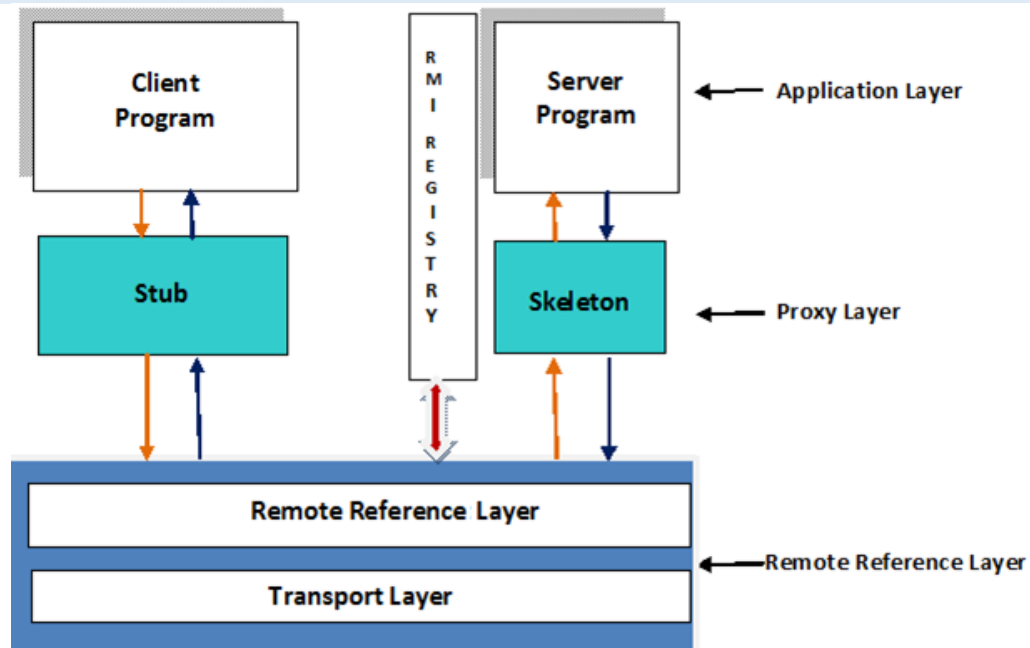


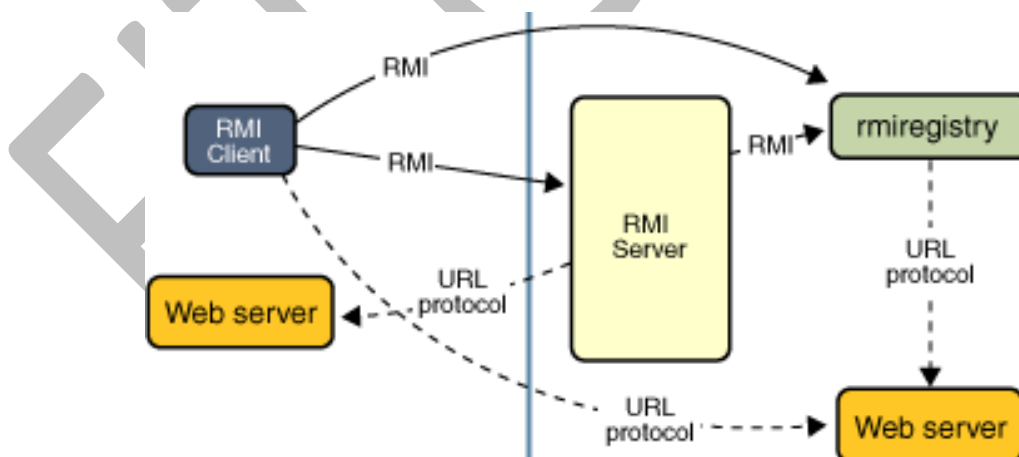
## INTRODUCTION

### RMI Architecture



### RMI Applications

The following illustration depicts an RMI distributed application that uses the RMI registry to obtain a reference to a remote object. The server calls the registry to associate (or bind) a name with a remote object. The client looks up the remote object by its name in the server's registry and then invokes a method on it. The illustration also shows that the RMI system uses an existing web server to load class definitions, from server to client and from client to server, for objects when needed.



(src: <https://docs.oracle.com/javase/tutorial/rmi/overview.html>)

## RMI EXAMPLE

### Bài 1

Hướng dẫn từng bước với bài toán xây dựng calculator server

1. Đầu tiên ta tạo một service interface. Interface này nằm ở cả 2 phía client và server, đóng vai trò như sự đảm bảo dịch vụ sẽ cung cấp từ server cho phía client cũng như là định nghĩa dịch vụ phải cài đặt của phía client.

Interface Calculator

```
public interface Calculator extends java.rmi.Remote{
    int cong(int a, int b) throws java.rmi.RemoteException;
    int tru(int a, int b) throws java.rmi.RemoteException;
    int nhan(int a, int b) throws java.rmi.RemoteException;
    int chia(int a, int b) throws java.rmi.RemoteException;
}
```

2. Lớp implements cài đặt dịch vụ sẽ phục vụ cho client

Lớp CalculatorImpl

```
public class CalculatorImpl
    extends java.rmi.server.UnicastRemoteObject
    implements Calculator{
    public CalculatorImpl() throws java.rmi.RemoteException{

    }
    public int cong(int a, int b) throws
java.rmi.RemoteException{
        return a+b;
    }
    public int tru(int a, int b) throws java.rmi.RemoteException{
        return a-b;
    }
    public int nhan(int a, int b) throws java.rmi.RemoteException{
        return a*b;
    }
    public int chia(int a, int b) throws java.rmi.RemoteException{
        return a/b;
    }
}
```

3. Lớp Server

Tạo một rmiregistry bind ở cổng 1099. Trong trường hợp tổng quát thì ta phải start **rmiregistry** như là 1 server riêng bên ngoài và sau đó bind dịch vụ của ta vào đó  
Tạo một service instance và đăng ký với server

```
import javax.naming.*;
import java.rmi.registry.LocateRegistry;

public class CalculatorServer{
    public static void main(String []args) throws Exception{

        //create local registry instead of using rmiregistry program
        LocateRegistry.createRegistry(1099);
        System.out.println("rmiregistry started on port 1099");

        //create implemnt instant
        Calculator calc=new CalculatorImpl();
        //bin to server - use JNDI
        Context ctx=new InitialContext();
        ctx.bind("rmi://localhost:1099/teo",calc);
    }
}
```

```

        //ctx.bind("rmi:met", calc);
        System.out.println("Service has bound to rmiregistry");
    }
}

```

#### 4. Lớp client

Tạo lớp client dùng **JNDI** lookup dịch vụ trên rmiregistry sao đó triệu gọi từ xa

```

import javax.naming.*;
public class CalculatorClient{
    public static void main(String []args) throws Exception{

        String svr="localhost";
        if(args.length>0) svr=args[0];

        //lookup reference using JNDI
        Context ctx=new InitialContext();
        Object obj=ctx.lookup("rmi://" +svr+":1099/teo");
        Calculator calc=(Calculator)obj;

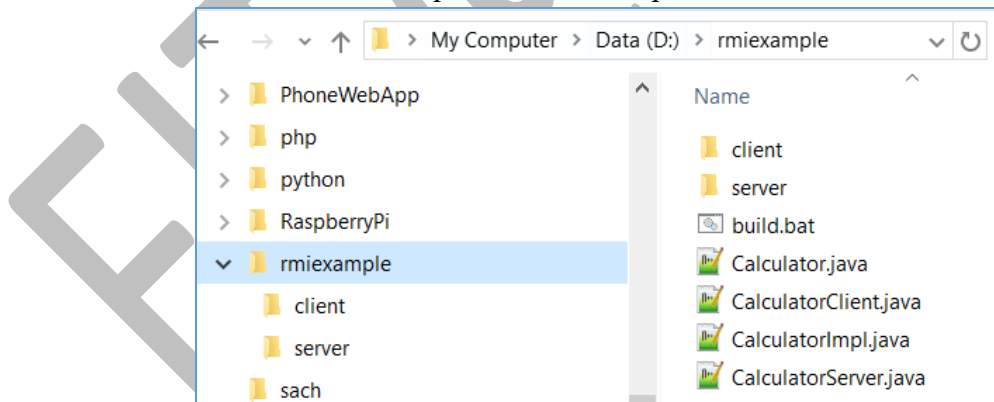
        //calling method
        int c1=calc.cong(3,6);
        int c2=calc.tru(3,6);
        int c3=calc.chia(3,6);

        //processing results
        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
    }
}

```

#### 5. Thực thi ứng dụng

Giả sử ta có thư mục rmiexample lưu trữ các lớp trên



a. Ta tạo file build.bat để build các file java

```

javac *.java
pause

```

Thực thi file bat này và đảm bảo không xuất hiện lỗi

b. Tạo 2 thư mục server và client trong thư mục này

Copy các file class vào các thư mục tương ứng:

- Calculator.class vào cả 2 thư mục client và server
- CalculatorImpl.class và CalculatorServer.class vào thư mục server
- CalculatorClient.class vào thư mục client.

c. Trong thư mục server tạo file server.bat có nội dung sau để start server

```
java CalculatorServer
```

## d. Trong thư mục client

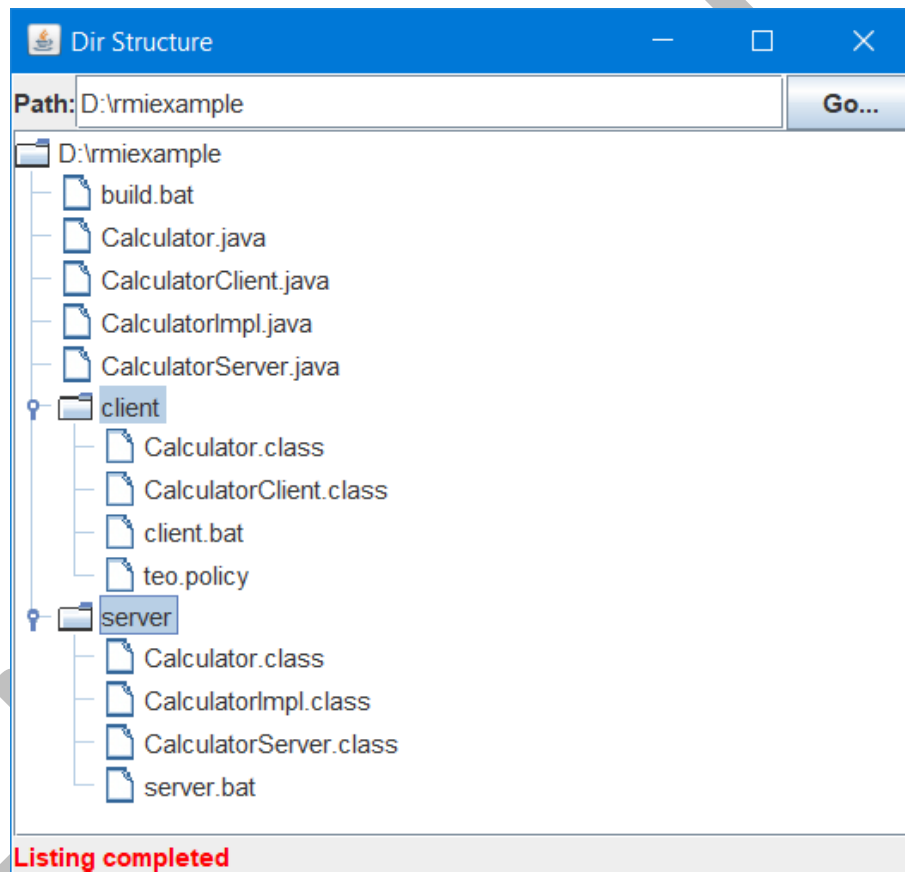
- Tạo file client.policy có nội dung sau

```
grant{  
    permission java.net.SocketPermission  
        "*:1024-65535", "connect,listen,resolve,accept";  
    //permission java.security.AllPermission;  
};
```

- Tạo file client.bat có nội dung sau

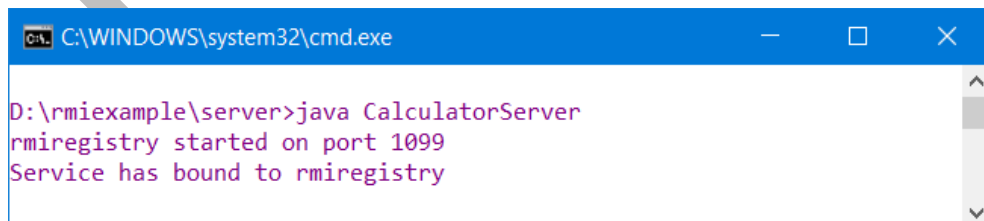
```
java -Djava.security.policy=teo.policy CalculatorClient  
pause
```

Sau khi tổ chức thư mục, tập tin. Cấu trúc tổng quát có dạng sau

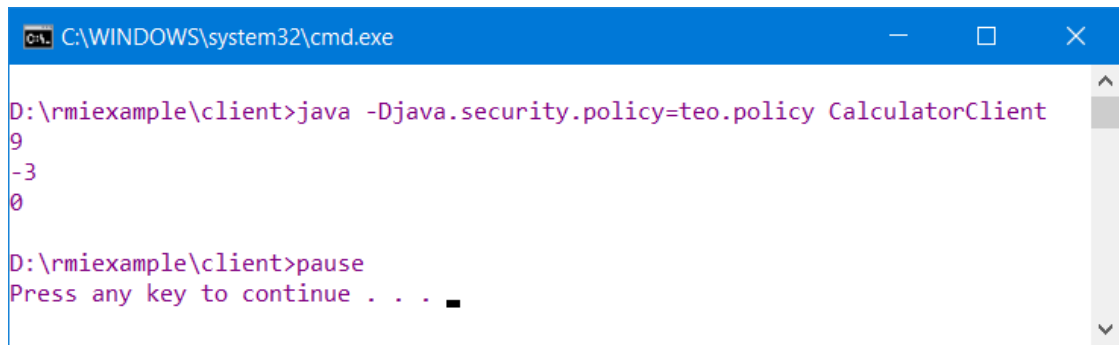


## 6. Thực thi và Kết quả

Server



Client

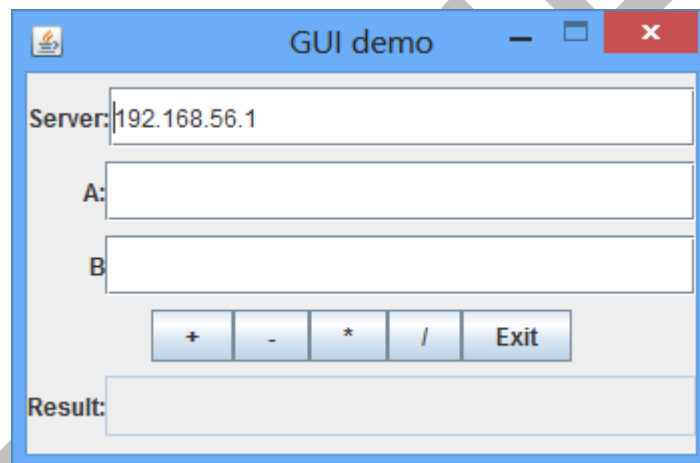


```
C:\WINDOWS\system32\cmd.exe

D:\rmiexample\client>java -Djava.security.policy=teo.policy CalculatorClient
9
-3
0

D:\rmiexample\client>pause
Press any key to continue . . . █
```

\*\*\* **Yêu cầu thêm:** Tạo client như sau sau đó kiểm tra kết nối trên máy khác để thực thi



## Bài 2

Mục tiêu: xây dựng ứng dụng dựa trên mô hình 3-tiers phân tán.

Yêu cầu: Xây dựng order-processing server cho phép các client tạo đơn đặt hàng từ xa.

## Entities

```
package entities;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Date;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class Order implements Serializable{
    private long _id;//order number
    private Date orderDate;
    private String customer;
    private String employee;
    private ArrayList<OrderDetail>details=new ArrayList<>();
    public Order(long orderNumber, Date orderDate, String customer,
String employee) {
        this._id = orderNumber;
        this.orderDate = orderDate;
        this.customer = customer;
        this.employee = employee;
    }
    public Order() {
    }

    public long get_id() {
        return _id;
    }
    public void set_id(long _id) {
        this._id = _id;
    }
    public Date getOrderDate() {
        return orderDate;
    }
    public void setOrderDate(Date orderDate) {
        this.orderDate = orderDate;
    }
    public String getCustomer() {
        return customer;
    }
    public void setCustomer(String customer) {
        this.customer = customer;
    }
    public String getEmployee() {
        return employee;
    }
    public void setEmployee(String employee) {
        this.employee = employee;
    }
    public ArrayList<OrderDetail> getDetails() {
        return details;
    }
    public void setDetails(ArrayList<OrderDetail> details) {
        this.details = details;
    }
}
```

```
@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + (int) (_id ^ (_id >>> 32));
    return result;
}
@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Order other = (Order) obj;
    if (_id != other._id)
        return false;
    return true;
}
@Override
public String toString() {
    GsonBuilder bd=new GsonBuilder();
    bd.setPrettyPrinting();
    Gson gs=bd.create();
    return gs.toJson(this);
}
}
```

```
package entities;

import java.io.Serializable;

public class OrderDetail implements Serializable{
    private String productName;
    private int quantity;
    private double price;
    public OrderDetail() {
    }
    public OrderDetail(String productName, int quantity, double price) {
        this.productName = productName;
        this.quantity = quantity;
        this.price = price;
    }
    public String getProductName() {
        return productName;
    }
    public void setProductName(String productName) {
        this.productName = productName;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {

```

```

        this.price = price;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((productName == null) ? 0 :
productName.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        OrderDetail other = (OrderDetail) obj;
        if (productName == null) {
            if (other.productName != null)
                return false;
        } else if (!productName.equals(other.productName))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return
productName+"\t"+quantity+"\t"+price+"\t"+(quantity*price);
    }
}

```

#### Data Access

```

package dataaccess;

import java.util.concurrent.CountDownLatch;
import org.bson.Document;
import org.bson.conversions.Bson;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.LongSerializationPolicy;
import com.mongodb.async.SingleResultCallback;
import com.mongodb.async.client.FindIterable;
import com.mongodb.async.client.MongoClient;
import com.mongodb.async.client.MongoClients;
import com.mongodb.async.client.MongoCollection;
import com.mongodb.async.client.MongoDatabase;
import entities.Order;

public class OrderDAO {
    MongoCollection<Document> col ;
    MongoClient client ;
    public OrderDAO() {
        client=MongoClients.create("mongodb://localhost:27017");
        MongoDatabase db = client.getDatabase("dhktpm11a");
        col = db.getCollection("orders");
    }
    public void addOrder(Order order) {
        try {

```



```

        CountdownLatch latch = new CountdownLatch(1);
        GsonBuilder gb=new GsonBuilder();

        gb.setLongSerializationPolicy(LongSerializationPolicy.STRING);
        Gson gs=gb.create();
        String js = gs.toJson(order);
        Document ds=Document.parse(js);
        col.insertOne(ds, new SingleResultCallback<Void>() {
            @Override
            public void onResult(Void result, Throwable t) {
                latch.countDown();
            }
        });
        latch.await();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

private Order order=null;
public Order findByOrderNumber(long orderNumber) {
    try {
        CountdownLatch latch = new CountdownLatch(1);
        Bson filter=new Document("_id", ""+orderNumber);
        FindIterable<Document> fi = col.find(filter);
        fi.first(new SingleResultCallback<Document>() {
            @Override
            public void onResult(Document result, Throwable
t) {
                if(t!=null)
                    System.err.println(t.getMessage());
                else {
                    Gson gs=new Gson();
                    order=gs.fromJson(result.toJson(),
Order.class);
                    latch.countDown();
                }
            }
        });
        latch.await();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return order;
}
}

```

## Business Logic

### Control

```

package businesslogic.controls;

import dataaccess.OrderDAO;
import entities.Order;

public class OrderControl {
    private OrderDAO dao;
    public OrderControl() {
        dao=new OrderDAO();
    }
}

```

```

    public void addOrder(Order order) {
        //???????? check validation
        dao.addOrder(order);
    }

    public Order findByOrderNumber(long orderNumber) {
        return dao.findByOrderNumber(orderNumber);
    }
}

```

*Facade*

Implement using RMI as façade

### 1. Service

```

package businesslogic.facade;

import java.rmi.Remote;
import java.rmi.RemoteException;
import entities.Order;

public interface OrderService extends Remote{
    //.....
    public void addOrder(Order order) throws RemoteException;

    public Order findByOrderNumber(long orderNumber) throws
    RemoteException;
}

```

### 2. Implementation

```

package businesslogic.facade;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import businesslogic.controls.OrderControl;
import entities.Order;

public class OrderServiceImpl extends UnicastRemoteObject
    implements OrderService{

    private OrderControl control;

    protected OrderServiceImpl() throws RemoteException {
        control=new OrderControl();
    }

    @Override
    public void addOrder(Order order) throws RemoteException {
        control.addOrder(order);
    }

    @Override
    public Order findByOrderNumber(long orderNumber) throws
    RemoteException {
        return control.findByOrderNumber(orderNumber);
    }
}

```

## 3. Server

```

package businesslogic.facade;

import java.rmi.registry.LocateRegistry;
import javax.naming.Context;
import javax.naming.InitialContext;

public class OrderServiceServer {

    public static void main(String[] args) throws Exception{

        LocateRegistry.createRegistry(1099);

        OrderService svc=new OrderServiceImpl();

        Context ctx=new InitialContext();
        String svr="localhost";
        if(args.length>0)
            svr=args[0];
        String url="rmi://" +svr+ "/ordersystem";
        ctx.bind(url, svc);
        System.out.println("Server is registered");
    }
}

```

## Client

```

package client;

import java.util.ArrayList;
import java.util.Date;
import javax.naming.Context;
import javax.naming.InitialContext;
import session15.businesslogic.facade.OrderService;
import session15.entities.Order;
import session15.entities.OrderDetail;

public class OrderClient {

    public static void main(String[] args) throws Exception{
        String svr="localhost";
        if(args.length>0){
            svr=args[0];
        }
        //lookup service interface
        Context ctx=new InitialContext();
        Object obj = ctx.lookup("rmi://" +svr+ "/ordersystem");
        OrderService service=(OrderService)obj;

        //create order
        Order order=new Order(System.currentTimeMillis(), new Date(),
            "Trương văn tũn", "Trần Thị MẾN");
        OrderDetail detail1=new OrderDetail("Xoài",10,100);
        OrderDetail detail2=new OrderDetail("Cóc",15,150);
        OrderDetail detail3=new OrderDetail("Ổi",100,200);
        ArrayList<OrderDetail>details=new ArrayList<>();
        details.add(detail1);
        details.add(detail2);
    }
}

```

```
        details.add(detail3);
        order.setDetails(details);

        //call rmi to insert object
        service.addOrder(order);
        System.out.println("Xong");

        //lookup order
        Order od = service.findByOrderNumber(1510663866348L);
        System.out.println(od);
    }
}
```

Triển khai và thực thi

## RMI ACTIVATION

### Bài 1

#### 1. Tạo service interface

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.io.Serializable;

public interface CalculatorActivable extends Remote, Serializable {
    public double calculate(double a, double b, char op) throws
    RemoteException;
}
```

#### 2. Tạo lớp implement

```
import java.rmi.MarshalledObject;
import java.rmi.RemoteException;
import java.rmi.activation.Activatable;
import java.rmi.activation.ActivationException;
import java.rmi.activation.ActivationID;

public class CalculatorImpl extends Activatable implements
    CalculatorActivable {

    protected CalculatorImpl(ActivationID id, MarshalledObject<?>
    data) throws ActivationException,
        RemoteException {
        super(id, 0 /*default port: 1099*/);
    }

    @Override
    public double calculate(double a, double b, char op) throws
    RemoteException {
        double kq=0d;
        switch (op) {
            case '+': kq=a+b; break;
            case '-': kq=a-b; break;
            case '*': kq=a*b; break;
            case '/': kq=a/b; break;
        }
        return kq;
    }
}
```

#### 3. Tạo lớp setup activator

```
import java.rmi.Remote;
import java.net.SocketPermission;
import java.rmi.activation.Activatable;
import java.rmi.activation.ActivationDesc;
import java.rmi.activation.ActivationGroup;
import java.rmi.activation.ActivationGroupDesc;
import java.rmi.activation.ActivationGroupDesc.CommandEnvironment;
import java.rmi.activation.ActivationGroupID;
import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;

public class CalculatorActivator {
```

```

public static void main(String[] args) throws Exception{
    //1. Install security manager for the ActivationGroup VM.
    System.setSecurityManager(new SecurityManager());
    //2. Set security policy
    Properties prop=new Properties();
    prop.put("java.security.policy", "server.policy");

    //3.Create an instance of ActivationGroupDesc class
    CommandEnvironment cmd=null;
    ActivationGroupDesc group=new ActivationGroupDesc(prop,
cmd);

    //4.Register the instance and get an ActivationGroupID
    ActivationGroupID
groupID=ActivationGroup.getSystem().registerGroup(group);

    //5.Create an instance of ActivationDesc
    //String location="http://localhost:8080/";
    String
location="file://C:/Users/VoVanHai/OneDrive/___Giangday/2017-2018-
hkl/JavaNC/lab/nw/ex1/";
    ActivationDesc desc=new ActivationDesc(groupID,
        CalculatorImpl.class.getName()/*implemented
class*/
        ,location/*location the object's code location
(from where the class is loaded)*/
        ,null);
    //6. Register the instance with rmid.
    Remote
calc=(CalculatorActivable)Activatable.register(desc);
    System.out.println("got the stub for CalculatorImpl");
    //7. Bind the remote object instance with its name
    Context ctx=new InitialContext();
    ctx.bind("rmi:calculator", calc);
    System.out.println("Exported from registration");
    //8. Exit the system
    System.exit(1);
}
}

```

#### 4. Client

```

import javax.naming.Context;
import javax.naming.InitialContext;

public class CalculatorClient {
    public static void main(String[] args) throws Exception{
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new SecurityManager());
        }

        Context ctx=new InitialContext();
        Object obj=ctx.lookup("rmi://localhost:1099/calculator");
        System.out.println(obj);
        System.out.println("-----");
        CalculatorActivable calc=(CalculatorActivable)obj;
        double kq=calc.calculate(5, 7, '+');
        System.out.println(kq);
    }
}

```

#### 5. Tạo các file policy

## a. server.policy

```
grant
{
    permission java.net.SocketPermission    "*:1024-65535",
    "connect,listen,resolve,accept";
    permission java.security.AllPermission;
};
```

## b. rmid.policy

```
grant
{
    permission com.sun.rmi.rmid.ExecPermission
    "${java.home}${/}bin${/}java";

    permission com.sun.rmi.rmid.ExecOptionPermission    "-
    Djava.security.policy=*";
};
```

## c. client.policy

```
grant
{
    //permission java.net.SocketPermission    "*:1024-65535",
    "listen,resolve";
    permission java.security.AllPermission;
};
```

## 6. Tạo các file bat

## a. 0. build.bat

```
javac -d . *.java
pause
```

Run file này và đảm bảo không còn lỗi.

## b. 1. rmid.bat

```
rmid -J-Djava.security.policy=rmid.policy
```

## c. 2. start rmiregistry.bat

```
rmiregistry
pause
```

## d. 3. run activation.bat

```
java -Djava.security.policy=server.policy CalculatorActivator
pause
```

## e. 4. client.bat

```
java -Djava.security.policy=client.policy CalculatorClient
pause
```

## 7. Thực thi

Thực các file theo thứ tự

## 8. Triển khai

Tiến hành tạo 2 thư mục client và server sao đó chép các file tương ứng của 2 phía vào. Triển khai server sau đó copy client sang các máy khác thực thi.

(Chú ý: sẽ phải chỉnh sửa lại địa chỉ máy chủ cho phù hợp việc thực thi máy bên ngoài)

## Bài 2

FT@IUH